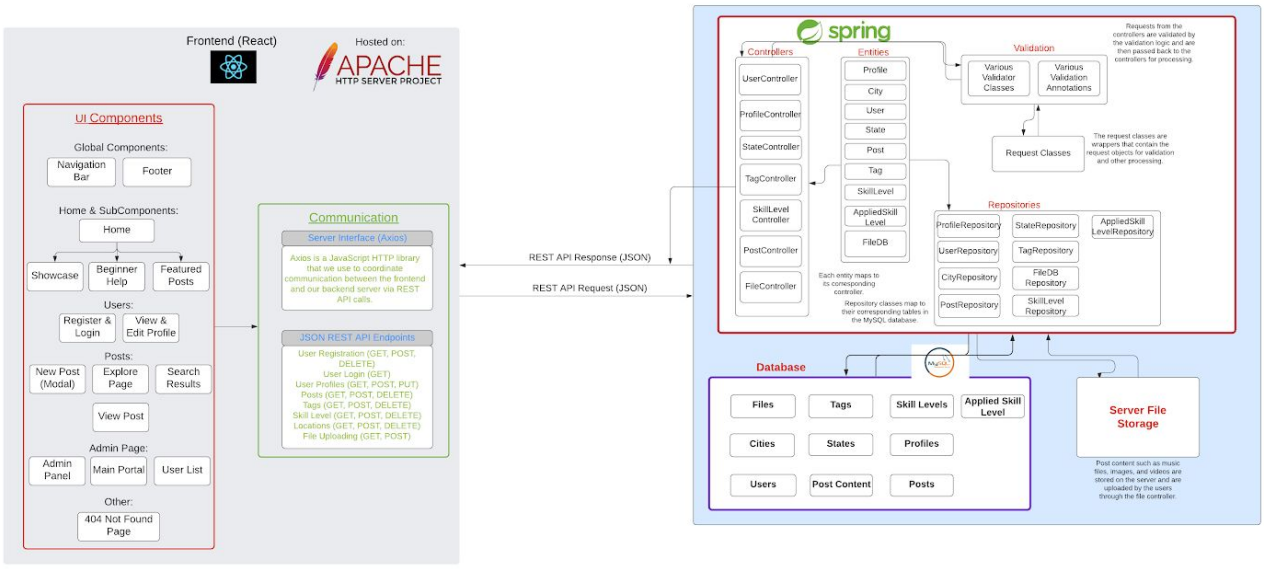
Block Diagram

BandMixer

CY_01

Tim Schommer, Andrew Groebe, Andrew Sandor,
Aidan Webster


Frontend (React)
Hosted on:
APACHE
HTTP SERVER PROJECT
UI Components
Global Components:
Navigation Bar
Footer
Home & SubComponents:
Home
Showcase
Beginner Help
Featured Posts
Users:
Register & Login
View & Edit Profile
Posts:
New Post (Modal)
Explore Page
Search Results
View Post
Admin Page:
Admin Panel
Main Portal
User List
Other:
404 Not Found Page
Communication
Server Interface (Axios)
Axios is a JavaScript HTTP library that we use to coordinate communication between the frontend and our backend server via REST API calls.
JSON REST API Endpoints
User Registration (GET, POST, DELETE)
User Login (GET)
User Profiles (GET, POST, PUT)
Posts (GET, POST, DELETE)
Tags (GET, POST, DELETE)
Skill Level (GET, POST, DELETE)
Locations (GET, POST, DELETE)
File Uploading (GET, POST)
REST API Response (JSON)
REST API Request (JSON)
spring
Controllers
UserController
ProfileController
StateController
TagController
SkillLevel Controller
PostController
FileController
Entities
Profile
City
User
State
Post
Tag
SkillLevel
AppliedSkill Level
FileDB
Each entity maps to its corresponding controller.
Repository classes map to their corresponding tables in the MySQL database.
Validation
Various Validator Classes
Various Validation Annotations
Requests from the controllers are validated by the validation logic and are then passed back to the controllers for processing.
Request Classes
The request classes are wrappers that contain the request objects for validation and other processing.
Repositories
ProfileRepository
StateRepository
AppliedSkill LevelRepository
UserRepository
TagRepository
CityRepository
FileDB Repository
PostRepository
SkillLevel Repository
Database
Files
Tags
Skill Levels
Applied Skill Level
Cities
States
Profiles
Users
Post Content
Posts
Server File Storage
Post content such as music files, images, and videos are stored on the server and are uploaded by the users through the file controller.

Backend Design Description:

Users:

The user entity and user controller are used for managing the list of users registered on our web service. The user controller is responsible for adding, removing, updating, and logging in our users. The user entity itself contains all necessary information for a user, including its ID number, username, email, and encrypted password.

Locations:

The location functionality is separated into two entities, states and cities. Each state has a list of cities associated with it, and each city has a state that it is attached to. The state controller is responsible for adding, removing, and updating the list of states in our system.

Posts:

The post functionality is heavily linked to the tagging system we have in place. The post entity contains all the information needed to construct a post on the front end. This includes the type of post, the content of the post, any media that the post contains, etc. Each post can also have a number of tags associated with it in order to facilitate searching. For instance a user can search for posts that are marked with tags “expert” or “guitar player” in order to easily find what they are looking for.

Profiles:

Profiles are simply an extension of posts. They are also able to have tags associated with them in order to facilitate easy searching. Profiles also contain contact information for the user, such as their phone number, as well as their location.

Tags:

The tagging system is the core of our web service. This is what enables users to easily search for potential band mates. All tags are stored in a tag database and are able to be accessed and applied to any post. As mentioned earlier, the tags are then used as a type of search filtering.

Frontend Design Description:

Website Framework:

We chose to create a website for our project's frontend. To achieve this, we are using **React.js** as our main framework, and **React-Bootstrap** as a CSS design framework. Using these two frameworks together has allowed us to create highly functional pages and website components that also look very good.

REST API Communication:

For communicating with our backend server's REST API framework, we are using a library called **Axios**. This is a very popular library used in many JavaScript website applications, and it allows us to create HTTP requests with corresponding methods (GET, POST, PUT, DELETE) and data (headers, body, endpoint routes, etc.).

UI Design:

As previously mentioned, we are using **React-Bootstrap** to develop our frontend UI design. This has allowed us to create great-looking pages and components without needing to "reinvent the wheel" by creating our own CSS styles. This framework comes with a great selection of pre-defined components, such as buttons, navigation bars, forms, image styling, etc,.

Component Design:

In typical React projects, website frontends consist of building blocks known as components. As it currently stands, our project consists of roughly 18 components, each representing a page or subsection of a page. React also enables developers to reuse components and define global components (used on all pages such as navigation bars and footers). We are taking advantage of all of these features as well.

Database Schema: